

**U.S. DEPARTMENT OF COMMERCE
National Technical Information Service**

PB-281 231

**Model State Information System Training
Program. Volume VI. Handbook for
System Specifications**

Cincinnati Univ, Ohio Southwestern Ohio Regional Computer Ctr

Prepared for

Environmental Protection Agency, Washington, D C

Jun 77

PB 281 231

Model State Information System
Training Program

Volume VI

Handbook

for

SYSTEM SPECIFICATIONS

Prepared for the

U.S. Environmental Protection Agency
Office of Water Supply
Washington, D.C.
under GSA Contract No. GS-05S-10458
Task Assignment OWS-77-01

by the

SouthWestern Ohio Regional Computer Center
Cincinnati, Ohio

June, 1977

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

BIBLIOGRAPHIC DATA SHEET		1. Report No. EPA 440/6-77-028-14	2.	3. Recipient's Accession No. PB281231
4. Title and Subtitle MSIS - Volume VI: Systems Specification Handbook				5. Report Date June 1977
7. Author(s) Southwestern Ohio Regional Computer Center				8. Performing Organization Rept. No.
9. Performing Organization Name and Address Southwestern Ohio Regional Computer Center 3333 Vine Street Cincinnati, OH 45220				10. Project/Task/Work Unit No. OWS-77-01
				11. Contract/Grant No. GSA Contract # GS-055-10458
12. Sponsoring Organization Name and Address U. S. Environmental Protection Agency Office of Drinking Water 401 M Street S.W. Washington, D.C. 20460				13. Type of Report & Period Covered
13. Supplementary Notes				14.
16. Abstracts This Handbook is one of six prepared for the U.S. Environmental Protection Agency's Office of Drinking Water by the SouthWestern Ohio Regional Computer Center. It was developed as part of installation planning to aid in training DP personnel about the Model State Information System (MSIS). It is intended to be used both by instructors and participants during the State installation of MSIS. Volume VI, System Specifications, covers basic design criteria and policies that governed the detail design, programming, testing, and documentation of MSIS.				
17. Key Words and Document Analysis. 17a. Descriptors				
17b. Identifiers/Open-Ended Terms				
17c. COSATI Field/Group				
18. Availability Statement RELEASE UNLIMITED		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 49	
		20. Security Class (This Page) UNCLASSIFIED	22. Price A43-A01	

ACKNOWLEDGEMENTS

No major effort may conclude without proper acknowledgement of help received. During the seven months required for gestation and birth of the six MSIS training courses, several individuals rendered indispensable help. Joseph Bagby of the University of Cincinnati's Faculty Resource Center, and Thomas Bartlett and Daun Richards of SWORCC bore the major burdens of creativity and revision. They were aided especially by Janis Cowperthwaite and Thomas Manz of SWORCC, who reviewed content for technical accuracy. Office of Water Supply personnel suffered many questions, especially Arnold Kuzmack, Fredrick T. Martin and Peter Bengtson. Finally, SWORCC's Systems Division Director, John W. Rust, provided urgently needed clerical and moral support. We all thank them all. To those whom we have inadvertently overlooked, our apologies and our thanks.

Ellen G. Miller
Project Manager, OWS-77-01

FOREWORD

The responsibility for implementing the Safe Drinking Water Act of 1974 was delegated to the Office of Water Supply (OWS) of the U.S. Environmental Protection Agency (EPA). In order to assist the States and EPA Regions with the capability to fulfill their basic responsibilities for monitoring water quality, as required by the Safe Drinking Water Act, OWS entered into a series of tasks.

The first series, with American Management Systems, Inc. (AMS) of Arlington, Virginia, resulted in both the concept of an informational system and its computerized version (the Model States Information System) which could be selected by interested States.

The second series of tasks was the SouthWestern Ohio Regional Computer Center (SWORCC) of Cincinnati, Ohio. One major effort was to perform the detail systems design, programming, testing, and documentation of the computerized Model State Information System (GSA Contract No. GS-05S-10458, Task Assignment No. OWS-76-02).

Installation Plan - Phase 1, the other major effort, was identified as Task Assignment No. OWS-77-01, under GSA Contract. The purpose of that was to assist the EPA Regions in ascertaining the needs of the States and Regions for aid while installing the MSIS and to develop training packages based on Regional recommendations.

The MSIS Training Program is a direct result of OWS-77-01. This handbook, one of two prepared as a guide for both instructors and participants on the contents of the system documentation created under Task Assignment OWS-76-02, is also a part of the overall MSIS Training Program which contains six separate volumes.

TABLE OF CONTENTS
Vol. VI System Specifications

	<u>Page</u>
ACKNOWLEDGEMENTS	11
FOREWORD	111
TABLE OF CONTENTS	v
PURPOSE	1
INTRODUCTION	2
DESIGN PHILOSOPHY	3
A. Modular	3
B. Chief Programmer Team (CPT)	5
C. Hierarchy plus Input/Process/ Output Diagrams (HIPOs)	5
D. Walkthroughs	10
E. Programming Support Library (PSL)	12
F. Documentation	13
PROGRAMMING PHILOSOPHY	24
A. Modular	24
B. Top Down Structured Programming	24
C. Standards	26
APPENDICES	39
A. Terms and Definitions	40
B. Reference Documents	42

PURPOSE

This Handbook is one of six prepared for the U.S. Environmental Protection Agency's Office of Water Supply (OWS) by the SouthWestern Ohio Regional Computer Center (SWORCC).

It was developed as part of installation planning to aid in training DP personnel about the Model State Information System (MSIS) also developed for OWS by SWORCC. It is intended to be used both by instructors and participants during the State installation of MSIS.

The installation of MSIS is accomplished through the joint efforts of SWORCC teams and the State's selected computer facility personnel. Without training, the computer facility would have a minimal knowledge of MSIS at the end of the installation process. To remedy this lack, Volumes V and VI of the MSIS Training Program were created.

There are six volumes of MSIS documentation, comprising 12 physical books. Volume V describes all 12 books, resulting in its being an annotated index to the system documentation, Volumes VII - XII.

This Volume VI, System Specifications, covers basic design criteria and policies that governed the detail design, programming, testing, and documentation of MSIS.

INTRODUCTION

There are two main considerations in developing a uniform system for a variety of different users. First, the system must be created and documented in such a way as to be easily understandable for the data processing organizations which will receive and be required to maintain and adapt the system to the specific requirements of their user organization. Second and in conjunction with the first, the system must be developed in such a way as to minimize the efforts of the receiving data processing organization when altering any part of the original basic system to the specific needs of its respective user organization.

As a result of these considerations the Model State Information System (MSIS) was developed utilizing a variety of relatively new data processing techniques collectively referred to as Improved Programming Technologies or IPT.

They consist of:

1. Chief Programmer Team (CPT) Operations;
2. Hierarchy plus Input/Process/Output (HIPO) Diagrams;
3. Top Down Structured Programming (TDSP);
4. Programming Support Libraries (PSL).

These techniques assure the attainment of several goals: Conformity to the general systems design, uniform modularity of the final system, a tested and easily maintainable system, and easily understandable, quality documentation.

Since IPT is a relatively new concept, the features are explained here for the data processing organization's understanding and use, if so desired, to maintain consistency within the system through subsequent changes. This document is intended to be used, therefore, by instructors and DP trainees for training at installation and later as a set of guidelines for the state data processing organization in developing modifications to MSIS.

DESIGN PHILOSOPHY

A. Modular

At the onset of the development of MSIS, American Management Systems, Inc. (AMS), of Arlington, Virginia, selected the modular system approach for MSIS and described it as follows in their Volume I: Subsystem Description, Definition of System Requirements, dated June 15, 1976.

"Under the new National Interim Primary Drinking Water Regulations, the states are expected to be responsible for performing a minimum set of functions. MSIS gives the states the capability to fulfill their responsibilities with some degree of variability. The system is not designed to satisfy all possible desires of all states, but it does provide a certain amount of flexibility in adapting the MSIS functions to individual states. The principal areas in which each state will have the ability to modify the system without changing the computer programs are:

- . The selection of what portions of a file will be used in creating the MSIS reports. Our system definition contains a flexible report selection language, enabling states to define the data to be contained in reports; however, report formats are predefined and cannot be changed.
- . The definition of what makes up the drinking water regulations. As long as the structure of the regulations remains unchanged, states may modify or add to the regulations table. For example, more chemicals can be analyzed, the state may define sampling requirements that are more stringent than the standard ones, the state may require a different MCL, etc.

"It must be emphasized, however, that any change in the structure of a file or in editing requirements will require program changes.

"Since the available computer hardware and software will differ among states, the MSIS is designed to be implemented using a standard programming language and primarily using sequential files that can be stored either on magnetic tape or disk.

"States also differ in the extent to which they already have ADP systems supporting their existing water supply programs. Even those states which already have an automated system for

managing drinking water quality data probably lack some of the functions of the MSIS. These states may want only portions of the MSIS. Therefore, the concept of a modular system has been carried over from the initial system concept, providing states with a choice regarding which subsystems of the MSIS they wish to implement. At the same time, care has been taken to use the same technical approach to file creation and maintenance for all subsystems. Error handling and update reporting has been standardized throughout the MSIS to facilitate the use of the same programming techniques in implementing the data entry and update modules for each subsystem. This approach should lead to efficient implementation, and, at the same time, will contribute to more efficient operations after the MSIS has been installed in a particular state." (p. 3-4)

The AMS philosophy and general design of the Model State Information System were accepted by the Office of Water Supply, which instructed SWORCC to use that general design as the foundation and guide for all subsequent detail design. SWORCC complied, sending draft versions of all detail design modules to OWS for review. In addition, OWS periodically convened Regional and State representatives to aid in the detail design. Of special assistance in that work was the MSIS Subcommittee of the Conference of State Sanitary Engineers.

The modular approach of AMS that resulted in subsystems and program modules was carried through in the detail design of MSIS. It facilitated development in that it allowed the scheduling of the most critical modules first and permitted the simultaneous design and programming of some modules while others were being programmed. One possible problem with the design and programming of a system by different individuals is the inconsistencies that could result due to the various backgrounds, experience, and training of the individuals performing the design. To avoid these inconsistencies, strict methods and standards were adopted. These methods and standards are the major topics of the remainder of this volume.

The modularity of the general system design was maintained during the detail design through the use of four basic techniques. First, Chief Programmer Teams were formed to develop the individual modules. Second, Hierarchy plus Input/Process/Output or HIPO Diagrams were used to illustrate, communicate, and document the functions of individual modules. Third, the consistency of modules was maintained through the use of Structured Walkthroughs both in design and programming. Finally, Programming Support Libraries were used to maintain consistency between all phases of design, programming, and testing.

B. Chief Programmer Teams

As a result of the modularity of the general design, a Chief Programmer was designated and assigned the responsibility for all the detail design and programming. Initially the Chief Programmer assigned each of the subsystems to teams of Senior Programmer(s) and Programmers for the detail design and programming; however, before any design or programming took place, the Chief Programmer Teams collectively reviewed the entire general systems design of MSIS. This was done so that everyone involved in the design had a clear understanding of the overall project and the interrelationships between the subsystems.

During the design and programming of MSIS, the Chief Programmer coordinated the operations to maintain consistency between similar functions, terminology, names, and standards.

C. Hierarchy plus Input/Process/Output: HIPO Diagrams

Hierarchy plus Input/Process/Output (HIPO) is a graphic tool which provides a consistent, visual, and functionally oriented method of approach to documentation.

- . HIPO s show graphically the project contents, the overview of the functions within the project, and the detail that describes the functions.
- . HIPO s stress function (what a program does) versus how it performs the function. This increases the initial understanding of a system, program, or segment because it visually presents functions. One is not required to rely on a complete narrative description.
- . HIPO s aid the maintainability of a system, program, or segment. A flowchart, for example, shows physical organization and logic. If a modification is necessary to the program logic, the flowchart can become outdated. HIPO s, on the other hand, show function. It is, therefore, less likely that a modification to program logic would affect a HIPO diagram.

The Senior Programmers utilized Hierarchy plus Input/Process/Output (HIPO) techniques as the primary means of communicating the detail systems design of the subsystems. HIPO's consist of three types of diagrams to functionally represent a system, subsystem, module, or routine. They are:

- . Visual Table of Contents (VTOC) Diagrams.
- . Overview Diagrams.
- . Detail Diagrams.

Under the direction of the Chief Programmer, the Senior Programmers performed the following steps for each subsystem:

- . Prepared a Visual Table of Contents Diagram for each subsystem of the MSIS. This diagram shows the function of the subsystem in its entirety with each subordinate function (a module) shown in relationship to other modules within the subsystem.
- . Prepared an Overview Diagram for each subsystem. This diagram defines all files and tables input to the subsystem, the major functions performed by the subsystem (what each module does), and all files and reports produced by the subsystem.
- . Performed a detailed analysis of each module in the nine MSIS subsystems.
- . Prepared a brief narrative of the function of each module.
- . Prepared an Overview Diagram for each module within a subsystem. This diagram defines all files and tables input to the module, the functional processes of the module, and the files and reports produced by the module.
- . Prepared a Detail Diagram for each functional process of the module. Detail diagrams, where possible, show specific input data elements required to perform the process, the functions of the processing steps, and the specific output data elements of the process. Subsequent lower level Detail Diagrams are provided as required to fully explain the function.

HIPO Standards - General

There are several levels of HIPO Standards used in MSIS beginning with these overall, general ones.

1. All numbering: Only give the number of the step's level is given (e.g., 1, not 9.2.1).
2. All entry and Exits: Entry and exit arrows point to the HIPO's before and after at the same level. The first

page of a level has no "in" arrow, the last page has no "out" arrow.

3. For all references to files in the input, process, and output section use only the abbreviation with the full name appearing once in the extended description. (Also, see item number 8.)
4. If an individual HIPO runs to more than one page, then it is indicated at the top of the page (e.g., 1 of 2, 2 of 2).
5. Each step in the Process section of an Overview or Detail has a matching line of description in the extended description. Each block on the first level of the VTOC has descriptive lines.
6. Where an input or output block appears more than once on an overview or detail, @1, @2, etc. are used to indicate multiple occurrences of the same file.
7. Only one verb is used per block plus 2 or three other words. Items with * are used only when necessary.

Standard verbs:

- (1) Update
- (2) Build
- (3) Select
- (4) Produce (=Print)
- (5) Schedule
- (6) Edit
- (7) Sort
- (8) Merge
- (9) Purge
- *(10) Verify
- *(11) Restore
- *(12) Delete

8. In the extended description, the full name of a file followed by the abbreviation in parenthesis is used the first time it is mentioned. Thereafter on that page, only the abbreviation is used.
9. Each subsystem has a VTOC and Overview.
10. Each block of the first level of the subsystem VTOC has an Overview. (Additional VTOC's and Overviews are used for clarity when needed.)
11. Each process step of an overview should have a Detail HIPO.

12. In the Input/Output section only the block symbol is used. No disk, tapes, or print symbols are used since the hardware will vary depending on the State.
13. Similar processes are described using standard wording whenever possible.

HIPO Standards - VTOC

There are three VTOC Standards:

1. The legend only appears on the VTOC. It contains:
 - a. Data movement
 - b. Entrys and Exits
 - c. @ -- Multiple I/O Reference
2. External sorts are not shown on VTOC's
3. The first level, in most cases, contains the modules of the subsystem.

HIPO Standards - Overview

The three overview HIPO Standards are:

1. Only permanent files and input transactions are indicated on Overviews. (Temporary files appear only on detail HIPO s or lower Overview diagrams.)
2. Only Program and HIPO references are given on Overviews. (Other references are reserved for detail HIPO s.)
3. External sorts are not shown. However, a note in the extended description on HIPO s should be used to explain what external sorts are needed.

HIPO Standards - Detail

Finally, the last level of standards are those for detail HIPOs. The Detail HIPO's provide as much information as is necessary for coding except where the volume of detail would have been prohibitive. In the Water Quality Compliance Subsystem (WAQC), appendices are used to add needed detail.

1. In the input/output boxes, all data elements which are used from a file are shown when possible. If there is not enough room to show all used data elements, do not show any of them. They will then be listed in the extended description. If a complete file is used, the individual data elements are not listed.
2. Edit criteria is only included in the extended description if it cannot be included in the Data Element Dictionary. Fields being edited are shown.
3. A description is included for each process step; the descriptions are numbered the same as the steps.
4. All inputs, outputs, and other documentation necessary to understand a step are referenced. The four letter subsystem identification is only used when referencing a different subsystem. Include references at the beginning of the step.
5. Input data will be referred to as "input transactions."
6. All printed output is referred to as "reports." The word "produce," not "print," is used when creating a report.
7. Temporary internal files are referred to as "internal."
8. All input/output is listed unless there is not enough room. Then a general term, such as "REPORTS", is used to indicate many reports. The full list is given in the "extended description."

The conventions for numbering forms, used under "REFERENCE" in the extended description, is as follows:

- a. Input forms: XXXX.100-199
- b. Input Record Layouts and Output Record Layouts: XXXX.200-299
- c. Output Reports: XXXX.300-399
- d. Turnaround Documents: XXXX.400-499

"XXXX" refers to a subsystem. Example: PWSI.102 would be a Public Water System Inventory Subsystem input form.

D. Walkthroughs

When a detail systems design package was completed and before it was finalized and submitted to the OWS for approval, a detailed review or structured walkthrough of the detail systems design for each defined module was performed to ensure the consistency, the quality, and the compatibility of the design with the other subsystems. The Chief Programmer Team (CPT) concept calls for structured walkthroughs which entail the mutual reading of the design code by other members of the team. The Chief Programmer, Project Manager, the Senior Programmer responsible for the design, and at least one other Senior Programmer participated in these reviews. When the design passed this review process, the final graphics were performed by the Project Librarian. The diagrams together with the associated documents then became the design specifications for use in coding the modules.

Likewise, when the coding for a program or function was completed, a detailed review or structured walkthrough of the program was also performed. This occurred before any testing began. The Chief Programmer, the Senior Programmer and/or Programmer who wrote the code, and at least one other Senior Programmer were present when the code was reviewed. In so doing, each member of the team reviewed code produced by other team members. This process verified the compatibility between individual program segments, and also permitted the identification of logic discrepancies or shortcomings before actual testing began. Each member of the team became extremely familiar with all parts of a given program, thus improving the subsequent testing and overall quality of the end product.

The purpose of walkthroughs is to recognize problems as early as possible. In a large system with many standards, few people can remember all details or can keep up with adjustments made in development or to standards. The walkthrough allows many minds to tackle a complex problem.

Procedures and Standards for Development Walkthroughs

Before the walkthrough:

1. The Team Leader coordinated a Walkthrough with the Chief Programmer.
2. The Team Leader also notified the individuals on that team and associated teams who were involved and distributed material at least one day in advance to all participants and other Team Leaders.

During the walkthrough:

1. The Team Leader stated the purpose of the Walkthrough to participants and solicited their overall comments about design, standards, etc.
2. The Team Leader also informally reviewed what was to be presented.
3. Then the group "walked through" the design, noting all specific comments.
4. The Librarian recorded major concerns, comments and decisions.

After the walkthrough:

1. Problems not corrected in the meeting were resolved and documented by the Chief Programmer.
2. The Librarian distributed a list of results from the meeting back to all participants and Team Leaders.

Procedures and Standards for Coding Walkthroughs:

1. The individual who wrote the program presented and explained their program.
2. The Team Leader reviewed the validity of the logic.
3. An MSIS COBOL expert reviewed the correctness and portability of the COBOL code.
4. The Librarian looked for common logic which should be copied or made a copy function.
5. What was coded for the initial walkthrough included:
 - a. IDENTIFICATION DIVISION
 - b. ENVIRONMENT DIVISION
 - c. DATA DIVISION
 - d. The first 3 levels of code-A,B,C levels:

A - DRIVER
B - INITIALIZE
B - PROCESS
B - TERMINATE

C levels controlling processing for each of the 3 B level paragraphs.

- e. Rest of paragraph names in program. Paragraphs always perform paragraphs at a lower level but not necessarily at the next lower level. Example: If a C and a D level paragraph perform the same paragraph it is at the E-level.
- f. A clean compile was necessary for the walkthrough.
- 6. Instructions for intermediate walkthroughs included (1) the coding for the paragraphs which were defined was filled in, and (2) the number of new lines of code to review was limited to a reasonable amount based on complexity and size.
- 7. The final walkthrough occurred when the complete program was reviewed. The JCL which was used to run the job, was included in this walkthrough.
- 8. After the final walkthrough, testing began.
- 9. When testing was considered complete and before the program was loaded out to the load library, a post-testing walkthrough took place to insure the quality of all test results and program code.

E. Programming Support Library (PSL)

The Programming Support Library (PSL) provides a means for the orderly development of computer programs.

- Unnecessary duplication of common logic is avoided by the use of the PSL. Common logic is coded and stored in the PSL. When it is needed, the logic is included in the subject program through the use of the COBOL COPY statement.
- The Project Librarian supports the programming process by performing many clerical tasks that are typically performed by individual programmers. Thus, staff utilization is optimized.
- The PSL establishes an exact correspondence between computer readable and human readable versions of the developing system. This guarantees on-going visibility of the developing system and permits standardization in project recordkeeping to ensure that hard copy listings correspond to the current version of the system.

F. Documentation

A complete set of documentation is provided to support both the detailed system design and the resulting software package of the Model State Information System. The following documents are supplied.

1. Program Documentation (Vol. XII)
2. User's Manual (Vol. VII)
3. Installation Manual (Vol. VIII)
4. Operations Manual (Vol. IX)

Finally, the portions of the Data Entry Instructions and Key punch Instructions were combined to form two more manuals, volumes XI and X, respectively.

During the design of each module, the Senior Programmer prepared HIPO's to illustrate the function(s) of the program. In addition, to communicate the whole design to the programmer the Chief Programmer Team prepared a brief narrative description of the functions, inputs, files, tables and outputs, as well as record layouts of each. They also prepared the appropriate portions of associated documents, such as Data Entry Instructions, Key punch Instructions, and Data Element Descriptions.

Thus, for each module defined in the general system design, a package of information within the Program Documentation is provided containing:

- A general narrative description of the purpose of the module.
- The Overview and Detail HIPO Diagrams for the module.
- Layouts of all input source documents, files, and tables, and output files and reports.

The following are outlines of these four manuals. (For detailed descriptions of all the MSIS manuals, see Vol. V, MSIS Components.

Volume XII, Program Documentation, includes the following:

a. Subsystem Documentation

- 1) Purpose
- 2) System Overview
- 3) Subsystem Descriptions

b. Data Element Dictionary

- 1) Purpose
- 2) Form Description
- 3) Catalogue Numbers
- 4) Index of Data Element Descriptions
- 5) Data Element Descriptions

c. Exhibits

- 1) Purpose
- 2) Form Description
- 3) List of Exhibits (by subsystem including input forms, record layouts, report layouts, and turnaround documents)

Volume VII, User's Manual, includes the following:

- a. Purpose
- b. Introduction
- c. User Defined Selection of Information
- d. User Defined Report Sequence
- e. Compliance Cycles
- f. Error Message Table
- g. Standard Error Messages for Processing Options
- h. Subsystem Descriptions (by subsystem including a general description, how to enter data, output reports, and error messages and actions)
- i. Summary of Data Requested to be Reported to EPA
- j. Summary of Data Required for MSIS Processing
- k. Appendix (including Data Entry Instructions Form Description, acronyms, and a glossary)

Volume VII, Installation Manual, contains all the data needed to implement the entire system. The manual contains the following:

- a. Purpose
- b. Minimum Hardware Configuration Requirements

- c. Minimum Software Requirements
- d. Installation (including tape characteristics, source modules, copy modules, job control modules, and files and tables)
- e. Installation

Volume IX, Operations Manual, has these sections:

- a. Purpose
- b. System Overview
- c. Descriptions of Program Modules
- d. Appendices. This includes the error message table of references.

The HIPO standards described earlier and other standards were utilized for the documents produced during the MSIS design. The seven major types of documentation standards, discussed in detail below, are:

- 1. Narratives
- 2. Data entry instructions
- 3. Key punch instructions
- 4. Data element descriptions
- 5. Input form format
- 6. Output form format
- 7. Naming

1. Narratives

Narratives are formatted as follows:

Subsystem: subsystem name
 Program Name: XXXX.XXX
 Date: date submitted to EPA or revised

I. Objectives:

(A brief paragraph describing the purpose of the module.)

II. Inputs:

- a. Record Forms and Layouts:

<u>NAME</u>	<u>RECORD NAME</u>	<u>EXHIBIT</u>	
		<u>FORM</u>	<u>LAYOUT</u>
(full file name)	(name on record layout)	(number)	(number)

(List the appropriate documents in II and III)

b. Processing Option Cards:

(Repeat same headings used in II.A. in II.B., III.A., III.B., and III.C.)

III. Outputs:

- a. Record Layout:
- b. Report Layouts:
- c. Turnaround Documents:

IV. Functions:

(A paragraph for each step in the module overview is numbered the same as the module steps.)

V. Associated Documents:

Input Record Layout

(The same as output record layout except as noted below)

- a. Permanent files are represented by output record layouts whether used as input or output. Input record layouts are used for temporary files and input data in some other form than that of the output record.
- b. The mnemonic name is not given for input layouts.
- c. The full name from the Output Layout is modified for input by adding an I to the three letter prefix. Example: SHF-FILE-ID becomes SHFI-FILE-ID. When adding an I makes the full name 31 characters, replace the F in the three letter prefix with the I. Example: SHF-FILE-ID... becomes SHI-FILE-ID....

Output Record Layout (Record Definition)

- a. "Output of Subsystem" is the subsystem creating the file.
- b. "Input to Subsystem(s)" are the subsystems using the file.
- c. "Date" is the date submitted to EPA or date updated.
- d. "Exhibit" is the identification number of the layout.
- e. "Record Type" is FIXED or VARIABLE. Fixed is used whenever possible.
- f. "Record Size" is the final record size. For variable length records give the minimum and maximum length of the record.
- g. "Page __ of __" shows pages of this form.
- h. "Record Access Key" gives the name(s) of the data element(s) which uniquely identifies a record.
- i. "File Name" - See Vol. XII, Program Documentation, Part 1 for file names.
- j. "File ID" - See Vol. XII, Program Documentation, Part 1 for file ID's.
- k. "Record Name" is the first name of the file. This is the file ID followed by "RECORD". Example: EAF-RECORD.
- l. "Level" - The same level numbers are used as found in the Cobol programs. Numbering starts with "01" and is incremented by 2 for the lower levels.
- m. "Data Element Mnemonic" - See the seventh documentation standard for Naming. The maximum length of any mnemonic is 10 positions. This field is only used for permanent MSIS files.
- n. "Data Element Name/Description" - See attached Naming Standards. The maximum length of any full data element name is 30 positions.

- o. "Data Element Catalogue Number" - Each data element on a file has a unique catalogue number. Those data elements which appear in two (or more) permanent files and have the exact same set of acceptable values on the 'MS' Code Value Table are edited against a common number. For all other data elements if a data element, such as Sampling Frequency, is on a file twice it must have two data element numbers in order that it can be uniquely identified during Standard History and Error Processing.
- p. "Data Type" - All data types are alphanumeric (A/N) if the data is never used in a mathematical calculation or is never be the result of a calculation.
- q. "Picture" - The Cobol format is used i.e., PICTURE X(9), PICTURE 9(5)V9(2). On input all dates are X(6); on output files all dates are 9(5).
- r. "Occurs" - Indicates the number of times an array of information occurs.
- s. "Var" - An 'X' is used here to indicate the beginning of a variable section of a record in a variable length record.
- t. "Position From - To" - These contain the start/end of an individual data element or the start/end of a group or an array of information.

2. Data Entry Instructions

- a. "Source Document Name" is the title of the input form.
- b. "Exhibit" is the input form number of the form which is being explained.
- c. "Purpose" is a sentence or two describing the purpose of the input form.
- d. "Legend" contains the abbreviations and definitions used in Additional Instructions." These are the same as those listed on the

Keypunch Instructions form but fewer are used. All numeric fields should be represented as right justified and preceded with zeroes.

- e. "Card Col. From/To" indicate the columns into which this information goes.
- f. "Input Document Name" is the heading of the field on the input form (or as close to this as possible where long headings are involved).
- g. "Description" provides instructions for entering this field.
- h. "Additional Instructions" are presented as abbreviations as listed on the Keypunch Instructions such as "N" for numeric.

3. Keypunch Instructions

- a. "Source Document Name" is the heading of the input form.
- b. "Exhibit" is the input form number of the form which is being explained.
- c. All other fields in the upper part of the form are for the user's use.
- d. "Card Col. From/To" are the column numbers to be punched.
- e. "Field Description: is the heading of the field involved.
- f. "Instructions/Keypunch" contains the appropriate abbreviations from the legend. (Example: A,LJ means Alpha, Left Justify)
- g. "Instructions/Other" is additional instruction needed for clarity.

4. Data Element Descriptions

- a. "Date" is the date submitted for EPA approval or the date last updated.
- b. "Version" is used when revisions are made.
- c. "Requested by EPA" tells whether the data element is needed for EPA reporting.
- d. "Mandated for MSIS" tells whether the data element is necessary for MSIS processing. Thus, a new record would not be created if this data element were missing.
- e. "Catalogue No" is the unique number associated with the data element.
- f. "Mnemonic Name" is the abbreviated (10 characters or less) name for the data element.
- g. "Program Name" is the extended (30 characters or less) name for the data element.
- h. "Description and Use" is a brief description of the data element and its use.
- i. "Data Type" is fixed (F) or variable (V).
- j. "Picture" is the COBOL standard picture of the data element.
- k. "Sources" is where the information for the creation of the data element comes from. Generally, this is an input form.
- l. "Input to" has those subsystems using the data element checked off.
- m. "Output of" has those subsystems producing the data element checked off. In most cases this is only one subsystem.
- n. "Acceptable Values" lists the acceptable values for the data element. If the acceptable values are on the Code Value Table (CVT) then the maximum number of acceptable values must be listed here.
- o. "Edit Criteria" lists restrictions on the values for the data elements and error codes resulting when edit criteria are not met.

- p. "Additional Comments" allow for additional explanation.

5. Input Form Format Standards

Forms contain the following headings:

- a. line 1: "Model State Information System."
- b. line 2: On the far left is the program number(s), of the program(s) which will use this input data; centered is the subsystem name.
- c. line 3: blank
- d. line 4 or more: Explanatory heading(s)
- e. one line blank
- f. detail headings
- g. one line blank
- h. detail lines are double spaced

Other format standards are:

- a. Any columns that are duplicated are indicated by the word "DUP" followed by the columns to be duplicated (Example: DUP 1-5).
- b. The following lines:

"Prepared by _____ Date / / / /

"Approved by _____ Date / / / /

are centered at the bottom of the form.

6. Output Form Format

- a. line 1: The first line of all MSIS reports and turnaround documents is:

"Model State Information System"

- b. line 2: The second line contains the seven position program number which generated the report in the first seven positions of the line. The Subsystem Name is centered on the line.
- c. line 3: One blank line.
- d. line 4: The date is printed in the first 13 positions of the line as DATE XX/XX/XX. An appropriate heading which describes the report or turnaround document is centered in the middle of this line. A page count is in the far right of this line, such as PAGE ZZZZ9.
- e. Subheadings are included as needed.
- f. At least 2 lines are skipped before starting the detail headings. All detail headings are as complete and meaningful as space permits.
- g. At least one line is skipped between the detail headings and the first line of the actual data.
- h. The number of lines printed on one page is limited to 56.
- i. When an error message is to be printed and a job terminated without any processing, the report headings which would have normally been produced are used with the error message printed below. Two exhibits are used with separate numbers; one with the error message and one of the normal report.
- j. Whenever totals are to be printed, follow the example of AMS Exhibit III.17 and put *'s in to identify clearly all totals as well as the relationship between one total and another.

7. Naming Standards

FILE IDENTIFICATION CODES

USE IN
MNEMONIC

USE IN
FILE ID

<u>NAME</u>	<u>FILE NAME</u>	<u>BLOCK</u>
PW	PUBLIC WATER SYSTEM INVENTORY FILE	(PWF)
PS	PERIODIC SAMPLE FILE	(PSF)
SA	SAMPLE AND ANALYSIS FILE	(SAF)
VI	VIOLATIONS FILE	(VIF)
RG	REGULATIONS FILE	(RGF)
VE	VARIANCE AND EXEMPTION FILE	(VEF)
EA	ENFORCEMENT ACTIONS FILE	(EAF)
OM	OPERATOR AND MAILING FILE	(OMF)
DV	DEVIATIONS FILE	(DVF)
SH	STANDARD HISTORY FILE	(SHF)
PH	PWS HISTORY FILE	(PHF)
PT	PUBLIC WATER TABLE	(PWT)
SE	STANDARD ERROR FILE	(SEF)
CS	COMPLIANCE STATUS FILE	(CSF)

OTHER TABLES

LB	LABORATORY TABLE	(LBT)
DE	DATA ELEMENTS TABLE	(DET)
CV	CODE VALUES TABLE	(CVT)
EM	ERROR MESSAGES TABLE	(EMT)

Mnemonic Naming

- a. The first two positions of each name is the 2 position File Identification Code followed by a dash (-).
- b. The other 7 positions are descriptive of the field.

EXAMPLE: PW-PLADDRE, for the first address line for the plant location.

Full Naming

- a. The first three positions of each name is the 3 position File Identification Code.
- b. Position 4 should be "-".
- c. Positions 5-30 is a descriptive name which clearly describes the data element.

PROGRAMMING PHILOSOPHY

A. Modular

The programming for each subsystem was done by a team consisting of the Senior Programmer(s) who designed the subsystem and by other Programmer(s) assigned to the subsystem. The Chief Programmer provided technical support to these teams as needed and maintained the direct responsibility for all work performed by each team.

The Senior Programmer(s) evaluated the complexity of the programs involved and determined the programs for which he was personally responsible for and which programs were to be done by the supporting Programmer(s). The Chief Programmer also coordinated this effort. Furthermore, the Chief Programmer coordinated the coding and use of common functions such as the calculation of the Julian date, so that unnecessary duplication of effort was avoided. The Project Librarian was responsible for storing these common functions in a library so that they were readily available when needed.

All programs were designed and coded to be as machine independent as possible using Top Down Structured Programming (TDSP) techniques in ANS COBOL. This is a technical programming approach which results in extremely modular programs which can be easily understood, facilitating the task of maintenance should an individual State desire to make any additions or modifications. The Chief Programmer and Project Manager established the TDSP standards which were followed in all programs. Such control of the programming process ensured the reliability, maintainability, readability, and the performance of each program.

B. Top Down Structured Programming

Top Down Structured Programming (TDSP) technology has its origin in Structured Programming (SP) which involves the writing of computer programs based on a mathematically proven structure theorem. The application of this theorem results in simpler, more maintainable programs because it reduces the number of logic structures which programmers use for writing programs. The permissible logic structures consist of only the following:

1. Sequences of two or more operations;

Example:

IF FIELD-A EQUAL TO FIELD-B	two
MOVE FIELD-C TO FIELD-D.	operations
IF FIELD-A EQUAL TO FIELD-B OR	more than
FIELD-A EQUAL TO FIELD-Q	two operations
ADD FIELD-R, FIELD-S GIVING FIELD-D	

2. Conditional branch to one of two operations and return;

Example:

```
IF FIELD-A EQUAL TO FIELD-B
MOVE FIELD-C TO FIELD-E
ELSE
MOVE FIELD-D TO FIELD-E.
```

3. Repetition of an operation;

Example:

```
PERFORM PARAGRAPH-1
VARYING INDX FROM 1 BY 1
UNTIL INDX EQUAL TO 25.
```

```
PARAGRAPH-1
ADD COUNT (INDX) TO TOTAL.
```

```
.
```

```
PARAGRAPH-2
```

A separate technique, which is an integral part of this technology, is Top Down Programming (TDP). This development technique results in a high degree of segmentation and permits detail design, coding, and integration of the source program code to proceed in parallel. It requires that the system be produced in a way which ideally eliminates writing of any code whose testing is dependent on other code not yet written, or on data which is not yet available.

Top Down Structured Programming (TDSP) improves program readability, understandability, maintainability, and reliability through the use of the following:

- Standard naming conventions for data names, variable names, and procedural paragraph names.
- One entry and one exit per program.

- A finite number of permissible logic structures (e.g., IF--THEN--else, repetitive execution of the same operation, etc.)
- Functional segmentation of code.
- Tree organization of structured segments in a top to bottom manner as is done in a departmental organization chart.
- Rules for language syntax (e.g., indentation, one verb per line of code, etc.)
- Comments and meaningful data names.

Test time is reduced by testing functional segments in the same manner as they are written, top down. Thus, continual integration of segments is possible because already tested segments drive the logic for subordinate segments.

C. Standards

Standards are those programming conventions that must be followed. Examples are:

- Every segment of code contains a single entry and a single exit.
- Explicit branching (GO TO type instruction) was not permitted except in certain rare cases.
- The beginning and end of any program or segment was completely contained within a single structured segment.
- Only one (1) verb per line of code was permitted.
- Indentation of source code statements was utilized to indicate the span of control of individual parts of the structured segment.
- Standard naming conventions were established for data elements, variable names, and procedural paragraph names.
- Features of Top Down Structured Programming were utilized.

- Program comments were inserted in-line as necessary for a clear understanding of the subject code.

Additionally, detail standards were established which relate to the use of the most efficient techniques available within the COBOL language.

Following are eleven additional standards used in the programming of MSIS.

1. Only use the ANS Cobol options which are acceptable in the Version 2 compiler. Do not use IBM extensions.
2. Only use the following Cobol logic control structures:

Nested if's (IFTHENELSE)

The pseudo code for a nested IF is as follows:

IF p

THEN

IF q

THEN

Statement-1

Statement-3

ELSE

Statement-2

Statement-3

ELSE

Statement-4.

Do not nest If statements more than 3 deep unless approved. Break up the If's or use a GO TO...DEPENDING ON.

3. Perform...Until (DOWHILE/DOUNTIL) examples:

Loop Structures: For multiple executions ending on a condition (zero or more times)

```
PERFORM loop-proc
UNTIL conditions.
```

For multiple executions ending on a condition (one or more times)

```
PERFORM loop-proc.
PERFORM loop-proc
UNTIL condition.
```

For multiple executions on a counter value (zero or more times)

```
PERFORM loop-proc
VARYING counter-name
FROM initial-value
BY increment
UNTIL condition
```

4. Do not use GO TO. Only in EXTREME cases, when determined by the Senior team leader that the maintainability of the program is improved by using a few GO TO's, will they be allowed. In some cases, PERFORM and GO TO DEPENDING ON can be used. For example:

```
PERFORM ACCOUNT-PROCESSING THROUGH ACCOUNT-
PROCESSING
-EXIT
.
.
.
ACCOUNT-PROCESSING.
GO TO
    IN-HOUSE
    CONTRACT
    SUB-CONTRACT
    NEW-BUSINESS
    SHOP-ORDER
    DEPENDING ON ACCOUNT-CODE.
error case processing statements.
GO TO ACCOUNT-PROCESSING-EXIT.

IN-HOUSE.
    processing statements.
    GO TO ACCOUNT-PROCESSING-EXIT.

CONTRACT.
```

processing statements.
GO TO ACCOUNT-PROCESSING-EXIT.

SHOP-ORDER.

processing statements.
GO TO ACCOUNT-PROCESSING-EXIT.

ACCOUNT-PROCESSING-EXIT.
EXIT.

5. Any portion of the Cobol program which is not considered "Portable" (easily transferable to another computer facility) is stored on the COPY LIBRARY and then copied into the program. This eases problems during installation. An example of this would be the Configuration Section of the Environmental Division. Common functions such as date manipulation, converting dates to Julian format and back, are also on the COPY LIBRARY and copied into all programs which require them. It is the responsibility of the Librarian to keep track of these functions.

6. Allowable Key Words are:

ACCESS	ELSE	MODE	SAME
ACTUAL	END	MOVE	SD
ADD	ENVIRONMENT	MULTIPLY	SEARCH
ADVANCING	EQUAL		SECTION
AFTER	ERROR	NEGATIVE	SELECT
ALL	EVERY	NEXT	SENTENCE
ALPHANUMERIC	EXAMINE	NO	SEQUENTIAL
AND	EXIT	NOT	SIZE
ARE		NUMERIC	SORT
ASCENDING	FD		SOURCE-COMPUTER
ASSIGN	FILE	OBJECT-	SPACE
AT	FILES	COMPUTER	SPACES
AUTHOR	FILE-	OCCURS	SPECIAL-NAME
	CONTROL	OMITTED	STANDARD
BEFORE	FILLER	ON	STOP
BLOCK	FIRST	OPEN	SUBTRACT
BY	FROM	OR	
		OUTPUT	TALLY
CHARACTERS	GIVING	PERFORM	TALLYING
CHECKING	GO	PICTURE	THAN
CLOSE	GREATER	POSITIVE	THEN
COBOL		PROCEDURE	THRU

COMMA	HIGH-VALUES	PROGRAM-ID	TO
COMPUTATIONAL	IDENTIFI-	QUOTE	UNTIL
CONFIGURATIONS	CATION	QUOTES	USE
CONTAINS	IF		USING
COPY	INDEXED	RANDOM	
CORRESPONDING	INPUT	READ	VALUE
	INPUT-	RECORD	VARYING
DATA	OUTPUT	RECORDING	
DATE-COMPILED	INSTALLA_	RECORDS	WHEN
DATE-WRITTEN	TION	REDEFINES	WORKING-
DECIMAL-POINT	INTO	RELEASE	STORAGE
DECLARATIVES	INVALID	REMAINDER	WRITE
DEPENDING	I-O	REMARKS	
DESCENDING	IS	REPLACING	ZERO
DIRECT		RETURN	ZEROS
DIRECT-ACCESS	JUSTIFIED	RIGHT	
DISPLAY	KEY	RUN	
DIVIDE			
DIVISION			
	LABEL		
	LAST		
	LEADING		
	LEFT		
	LESS		

LINES

LOW-VALUES

7. Format Summary

(Items marked with an * are copied into programs.)

IDENTIFICATION DIVISION.

PROGRAM-ID. program-name

AUTHOR. SouthWestern Ohio Regional Computer Center

INSTALLATION. EPA -- Office of Water Supply.

DATE-WRITTEN. month and year.

DATE-COMPILED. (machine generated)

REMARKS. (comment entry which explains purpose
and general functions of the program.
Copier cannot be used in the Identifi-
cation Division.)

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

*SOURCE-COMPUTER COPY SCOM.

*OBJECT-COMPUTER COPY OCOM.

INPUT-OUTPUT SECTION.

*FILE-CONTROL.

SELECT file name COPY assign statement

*SPECIAL-NAMES. COPY

DATA DIVISION.

FILE SECTION.

*FD File name COPY file copy name
or for work files
FD file name
RECORD CONTAINS integer (TO integer) CHARACTERS
BLOCK CONTAINS integer RECORDS
LABEL RECORDS ARE STANDARD
OMITTED
DATA RECORDS ARE record name (Include in the FD
RECORD IS statement but not in the FD
copy.)
*01 record name COPY record copy name.

(The print record must include 133 positions, X(1) and X(132), but only print 132 characters. Print the 132 position field by moving them to the print record.)

WORKING-STORAGE SECTION.

77 FILLER PICTURE X(30) VALUE program name
WORKING STORAGE BEGINS (or ENDS)
Put this 77 level at the beginning of the
Working Storage Section.
01-49 data name PICTURE
FILLER (USAGE) INDEX
REDEFINES COMPUTATIONAL
JUSTIFIED RIGHT (LEFT)
VALUE literal
88 condition-name
VALUE literal-1 (THRU literal-2)
VALUES
OX FILLER PICTURE X(30) Value program name
WORKING STORAGE ENDS. Put this at the
end of the Working-Storage Section.

(Group all switches, counters, hold areas, etc. together.) (Do not use underlines or dashes.)

PROCEDURE DIVISION.

ADD identifier-1 (identifier-2 TO identifier-n
(identifier-m) (ON SIZE ERROR
imperative-statement)
ADD CORRESPONDING identifier-1 TO identifier-2
(ON SIZE ERROR imperative statement)
CLOSE file-name-1...file-name-n
DECLARATIVES.
END DECLARATIVES.
DISPLAY literal-1 (identifier-1)
DIVIDE identifier-1 BY identifier-2

(ON SIZE ERROR imperative-statement)
DIVIDE identifier-1 BY identifier-2
GIVING identifier-3
 (REMAINDER identifier-4)
 (ON SIZE ERROR imperative-statement)

EXAMINE identifier REPLACING ALL
LEADING
FIRST
UNTIL FIRST
 literal-1 BY literal-2

EXIT.
GO TO procedure-name-1
GO TO procedure-name-1...-n DEPENDING ON
 identifier

IF condition THEN statement-1
NEXT SENTENCE
ELSE statement-2
NEXT SENTENCE

MOVE identifier-1 TO identifier-2...
 identifier-n

MULTIPLY identifier-1 BY identifier-2 GIVING
 identifier-3 (ON SIZE ERROR
 imperative-statement)

OPEN (INPUT file-name)(OUTPUT file-name)
 (I-O file-name)

PERFORM procedure-name-1(THRU procedure-
 name-2)

PERFORM procedure-name-1
PERFORM procedure-name-1
UNTIL condition-1
PERFORM procedure-name-1

VARYING index-1 FROM index-2
BY literal-3
 identifier-1 identifier-2

UNTIL condition-1
READ file-name (INTO identifier)
AT END imperative statement
INVALID KEY

STOP RUN (A program should contain only
 one stop run.)

SUBTRACT identifier-1 (identifier-2) FROM
 identifier-m identifier-n
 (ON SIZE ERROR imperative statement)

SUBTRACT identifier-1 (identifier-2...) FROM
 identifier-m GIVING identifier-m
 (ON SIZE ERROR imperative statement)

SUBTRACT CORRESPONDING identifier-1 FROM
 identifier-2

(ON SIZE ERROR imperative statement)
WRITE record-name (FROM identifier-1)
 (AFTER ADVANCING integer LINES)
WRITE record-name (FROM identifier-1)
INVALID KEY imperative-statement

Internal Sort Considerations

SD sort-file-name
 RECORD CONTAINS integer (TO integer)
 CHARACTERS DATA RECORD IS record name
 01 record name
 03 prefix-SORT-KEY PICTURE X(30)
 .
 .
 .

RELEASE sort-recrod-name (FROM identifier)
RETURN sort-file-name (INTO identifier)
 AT END imperative statement
SORT sort-file-name ON ASCENDING KEY prefix
 SORT-KEY
 INPUT PROCEDURE IS section-name-1(THRU
 section-name-2)
 USING file-name-1
 OUTPUT PROCEDURE IS section-name-1
 (THRU section-name-2)
 GIVING file-name-2

Table Handling Considerations

SEARCH identifier (VARYING index-1
 identifier-2
 (AT END imperative-statement-1)
 WHEN condition-1 imperative-statement-2
 NEXT SENTENCE
SET index-name TO identifier-1
 UP BY literal-1
 DOWN BY
SEARCH ALL identifier-1 (AT END
 imperative-statement-1)
 WHEN condition-1 imperative-statement-2
 NEXT-SENTENCE

8. In general all names should be as meaningful as possible.

a. File Names

- 1) Permanent Files - the name consists of the 3 digit file ID, followed by a hyphen, followed by the word FILE. All programs that use a file must use the same name.

EXAMPLE - PWF-FILE

- 2) Temporary Files - the name consists of the work TMP-, followed by some file identification, followed by a hyphen, followed by the word FILE. EXAMPLE - TMP-CONDENSED-FILE.

- b. Paragraph Names - the name of each paragraph (segment) consists of a letter, (starting with A), followed by a sequence number, (starting with 01 and increasing by 2), followed by a hyphen, followed by a meaningful description of what the paragraph's function is. Use the function description from the HIPO's, i.e. the verb plus description.

Thus, the first control segment for a program is a single A-paragraph whose name is A01 Driver. This segment performs three B-level paragraphs:

B01 - Initialize
B03 - Process
B05 - Terminate

These B-level paragraphs perform lower level paragraphs and contain little executable code themselves. Each performed paragraph will then be a lower level but not necessarily at the next lower level. For example if a C-level paragraph and a D-level paragraph perform the same paragraph, the performed paragraph would be at the E-level (one level lower than the lowest level paragraph performing it). Thus, the paragraph naming and actual physical placement of each performed routine is determined by reason for coding the PERFORM.

Structured code does not allow for one paragraph to fall through to another. Every paragraph should have a reference except the beginning of the program or the beginning of a section.

Working Storage Variables - the name should consist of the letters WS for all level 77 and 88 variables and the letter W for all variables in an 01 level group, followed by a sequence number, (starting with 01 and incrementing by 2), followed by a hyphen, followed by a meaningful description of the variable. If appropriate, variables

are grouped under an 01 level rather than making level 77 entries. 98 level numbers are used to aid in documenting programs. Examples: 77 WA01-switch, 01 W03-HEADER.

Structures

- 1) Start the Level/number (01) in card column 8.
- 2) Increment level numbers by 2: 03, 05, etc.
- 3) Starting with the '03' Level, put the next lower level number in the next two card columns to the right of the previous Level number.

```
8      12 14
01 RECORD.
      03 REC-TYPE
          05 REC-CODE
          05 REC-NUM
              07 REC-DIG1
              07 REC-DIG2
```

- 4) Always code the complete word "PICTURE" and put it in CC 52-58. Then skip 1 space before defining the picture specifications.

- a) Use the symbol "X" for alphanumerics.
- b) Use the symbols "9", "S", and/or "V" for numerics and put "USAGE COMPUTATIONAL" whenever it is appropriate to use on the line directly below the picture statement.
- c) Use the repetition capability in defining picture attributes. For example use X(3), not XXX; use X(1), not X.
- d) If a value clause is needed, then put it directly under the variable name.

```
01      RECORD
      01      NUMBER
          VALUE +55
```

- 5) When naming the variables in a structure used for producing a report, precede the variable names with "REPT-" unless using the name "FILLER".

```
01      REPT-HEADER-1.
      03      REPT-TITLE
      03      FILLER
```

03 W#-REPT-DATE

- 6) When naming the variables in a structure used for processing options, precede the variable names the "W#-PROC-" unless using the name "FILLER".

NOTE: As with structures, when defining variables in Working Storage line up the Picture Clauses, Value Clauses, Names, etc. to make this section easy to read.

9. Indentation

Statements between IF and ELSE should be indented 4 spaces and coded as follows:

IF condition

statement

IF condition

statement

ELSE

ELSE

Have all level 01 and level 77 have their level numbers in columns 8-9 and their names start in column 12. Indent condition names (88 levels) 2 columns from the condition variables.

Any time more than one line is needed for a statement or entry, indent the continuation 4 spaces to make it stand out.

Use 1 blank line to separate paragraphs, structures, etc. to make their relationship more apparent.

Important Clauses for Verbs should be set off by writing them on separate lines and indenting them 4 spaces.

PERFORM LINE-OUT

VARYING LINE-NUMBER FROM 1 BY 1

UNTIL LINE-NUMBER IS GREATER THAN 50.

Align similar statements to make them consistent.

OPEN INPUT TRANSACTION-FILE

OLD-MASTER-FILE

OUTPUT EXCEPTION-FILE

NEW-MASTER-FILE

REPORT-FILE

MOVE T-ACCOUNT-NUMBER TO R-ACCOUNT-NUMBER.

MOVE T-HOURS-WORKED TO R-HOURS-WORKED.

MOVE T-PAYRATE TO R-PAYRATE.

MOVE W-GROSSPAY TO R-GROSSPAY.

MOVE W-NETPAY TO R-NETPAY.

10. Comments

Put 1 line of asterisks and 1 blank line before and after every note.

Always put a note in front of each paragraph. It should immediately precede the paragraph it is intended to describe.

A note should be used to explain WHAT the code is doing, not how.

11. Procedure Division

Each paragraph should have one entry and one exit.

A paragraph name should be on a line by

Only one verb per line of code.

Do not use operational symbols in your code. Use the complete words. - ADD, etc. (+, -, *, etc.)

Do not use any abbreviations of Key words.

Do not have any literals in the Procedure Division with the only exception being adding or moving a numeric literal to a counter, switch, etc. Define values in Working Storage. (This also applies to Data Element Catalogue Numbers.)

Keep the size of a single paragraph to between 50-70 lines of code. Try to make your paragraphs relate back to the functions on the HIPO's.

Structure your code - high level control paragraphs at the beginning, detail paragraphs following in order as shown on HIPO's.

The first statement in a paragraph should begin in column 12.

For printing the heading at the top of a page, use WRITE record name (FROM identifier-1) AFTER ADVANCING NEW-PAGE where NEW-PAGE is a variable properly defined and copied into the SPECIAL-NAMES section of ENVIRONMENT DIVISION.

Use () whenever there are compound conditions (and's and or's). This will prevent confusion on the sequence of processing.

APPENDICES

APPENDIX A
Terms and Definitions

CPT -- Chief Programmer Team.

Chief Programmer -- The individual responsible for all design, programming, testing/debugging, and documentation. The senior member of the Chief Programmer Team.

Chief Programmer Team (CPT) -- A group of individuals who work closely together in the generation program code, testing, and documentation.

Functional Segmentation -- The process of segmenting logic based on the function for which the code was generated.

Hierarchy plus Input/Process/Output (HIPO) -- A technique which serves as a design and documentation tool. It increased understanding by showing the function of a system, program, or segment through the use of a standard graphic syntax.

HIPO -- Hierarchy plus Input/Process/Output.

Improved Programming Technologies (IPT) -- A term used to collectively refer to state-of-the-art data processing technologies. Included are Hierarchy plus Input/Process/Output, Top Down structured Programming, Programming Support Library, and Chief Programmer Team Operations.

IPT -- Improved Programming Technologies.

Programming Support Library (PSL) -- A repository for data necessary for the orderly development of computer programs using Improved Programming Technologies. PSL contains standard file descriptions and logic segments which are common to more than one program.

Project Librarian -- An individual who has responsibility for maintaining the technical records of the Programming Support Library; a member of the Chief Programming Team.

PSL -- Programming Support Library.

SP -- Structured Programming.

Structured Program -- A program constructed of a basic set of control logic structures which provide at least the following: Sequence of two or more operations,

conditional execution of one of two operations and repetition of an operation. A Structured Program has only one entry and one exit point. In addition, a path always exists from the entry to each subordinate segment and from each subordinate segment to the exit.

Structured Programming (SP) -- The process of developing Structured Programs. Associated with Structured Programming are certain practices such as indentations of source code to represent logic levels, and the use of intelligent data names and descriptive commentary.

Structured Segment -- A collection of functionally related logic that makes up a complete set of executable instructions.

TDP -- Top Down Programming.

TDSP -- Top Down Structured Programming.

Top Down Programming (TDP) -- The concept of performing in hierarchical sequence a detailed design, code, integration, and test as concurrent operations.

Top Down Structured Program -- A Structured Program with the additional characteristics of the source code being logically, but not necessarily physically, segmented in a hierarchical manner and only dependent on code already written. Control of execution between segments is restricted to transfers between adjacent hierarchical segments.

Top Down Structured Programming (TDSP) -- The process of developing top down structured programs. Associated with Top Down Structured Programming are certain practices such as indentations of source code to represent logic levels, the use of intelligent data names, and descriptive commentary. Top Down Structured Programming requires Top Down Programming as the primary implementation methodology.

Reproduced from
best available copy.



APPENDIX B
Reference Documents

The following documents were used during the MSIS detail design phase:

COBOL, FIPS Pub 21-1, Category: Software; Subcategory: Programming Languages. December 1, 1975, U.S. Department of Commerce, National Bureau of Standards.

Data Structuring Study, Vol. IV, April 1975, Structured Programming Series, RADC-TR-74-300. International Business Machines Corporation, Federal Systems Center, produced under U.S. Air Force (RADC) Contract F30602-74-C-0186, co-sponsored by U.S. Army Computer Systems Command.

Design Document: Definition of System Requirements, American Management Systems, Inc. April 15, 1976, produced under U.S. Environmental Protection Agency Contract 68-01-1939 for the Office of Water Supply.

Documentation Standards and Requirements, Chapter 6, Appendix: C - System/Subsystem Design Specification Guidelines; Appendix D - User Manual Standards; Appendix E - Systems Maintenance Documentation Standards. U.S. Environmental Protection Agency, Automatic Data Processing Manual, March 31, 1975.

Estimating Software Project Resource Requirements, Vol. XI, January 1975, Structured Programming Series, RADC-TR-74-300. International Business Machines Corporation, Federal Systems Center, produced under U.S. Air Force (RADC) Contract F30602-74-C-0186, co-sponsored by U.S. Army Computer Systems Command.

Flowchart Symbols and Their Usage in Information Processing, FIPS Pub 24, Category: Software; Subcategory: Documentation. June 30, 1973, U.S. Department of Commerce, National Bureau of Standards.

HIPO: Design Aid and Documentation Tool, First edition (April 1973), Copyright IBM Corp., 1973. International Business Machines Corporation, Data Processing Division.

Inmon, Bill, "An Example of Structured Design," Datamation, March 1976, Vol. 22, No. 3, page 82.

Jones, K.D., "Top-Down Approach Necessary for Structured Program Clarity," April 26, 1976, Vol. X, No. 17, page 26.

Jones, Martha Nyvall, "HIPO for Developing Specification." Datamation, March 1976, Vol. 22, No. 3, page 112.

Program Design Study, Vol. VIII, May 1975, Structured Programming Series, RADC-TR-74-3000, International Business Machines Corporation, Federal Systems Center, produced under U.S. Air Force (RADC) Contract F30602-74-C-0186, co-sponsored by U.S. Army Computer Systems Command.

Program Evaluation Requirements, Concepts, and Plans for the EPA Safe Drinking Water Program, Volumes I-III. American Management Systems, Inc., January 12, 1976, produced under U.S. Environmental Protection Agency contract 68-01-1939 for the Office of Water Supply.

Programming Language Standards, Vol. I, March 1975, Structured Programming Series, RADC-TR-74-300. International Business Machines Corporation, Federal Systems Center, produced under U.S. Air Force (RADC) Contract F 30602-74-C-0186, Co-sponsored by U.S. Army Computer Systems Command.

Software Summary for Describing Computer Programs and Automated Data Systems, FIPS Pub 30, Category: Software; Subcategory: Documentation. June 30, 1974, U.S. Department of Commerce, National Bureau of Standards.

Test and Debugging Aids, Datapro Directory of Software, Section: D80; Title: Programming Aids. Copyright Datapro Research Corporation, 1976, page D80-500-001.

Validation and Verification Study, Vol. XV, May 1975, Structured Programming Series, RADC-TR-74-300. International Business Machines Corporation, Federal Systems Center, produced under U.S. Air Force (RADC) Contract F30602-74-C-0186, co-sponsored by U.S. Army Computer Systems Command.

van Wolferen, Anthony, "GOTO 'Hopscotch' Not Structured," April 26, 1976, Vol. X, No. 17, page 26.